EXACT WHERE IT MATTERS: DOWNSTREAM-GUIDED ATTENTION REFINEMENT FOR LONG-VIDEO UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Approximating attention accelerates long-video understanding, but local error alone does not determine whether a correction improves the answer. We introduce EWiM (Exact Where it Matters), a training-free method for downstream-guided attention refinement during offline video–question prefilling. At selected layers, EWiM probes a bounded candidate pool through isolated corrections on sampled query rows and measures their effects on a question-conditioned next-layer readout at the terminal prompt position. Incremental updates reuse the common coarse state and refresh only the changed contributions to the next-layer readout. An execution-aware allocator uses these one-step sensitivity scores and hardware-profiled marginal costs to select full-block refinements, updating costs to account for shared row computation within a budget that includes probing overhead. Input tokens, model parameters, and the decoding procedure remain unchanged. Across eight main model–benchmark pairs, EWiM retains 99.2% of dense accuracy on average at a $1.46\times$ geometric-mean speedup in time to first token, including visual encoding and all acceleration overhead. Controlled interventions show that the selected block sets yield higher signed answer-margin utility than those chosen by local-error controls. Further experiments demonstrate transfer to additional backbones and composition with token compression.

1 INTRODUCTION

Recent Video Large Language Models (Video LLMs) have shown strong capabilities in long-video understanding, enabling models to reason over increasingly long and information-dense visual contexts (Zhang et al., 2024; Shen et al., 2024; Li et al., 2024a; Maaz et al., 2024; Zhang et al., 2025; Bai et al., 2025b;a). However, the resulting visual sequences can contain thousands of tokens, making LLM prefilling a major efficiency bottleneck (Li et al., 2025b). Existing acceleration methods mainly follow two paths: visual token compression shortens the input sequence by pruning or merging redundant tokens (Shen et al., 2025; Fan et al., 2026), while sparse or approximate attention keeps the tokens but reduces the cost of their interactions (Li et al., 2025b;a).

Sparse and approximate attention methods have moved from hard sparsification toward more flexible approximation. Some methods exploit structured attention patterns to skip selected interactions, while others preserve coarse contributions through pooled or multi-resolution key–value representations (Li et al., 2025b; Liu et al., 2026; Li et al., 2025a). Recent work further treats exact computation as a limited resource and selectively restores it for interactions whose approximation appears unreliable (Zhou et al., 2026). These methods suggest a coarse-to-refine paradigm: compute most interactions cheaply, then revisit a small subset with exact attention.

However, the coarse-to-refine paradigm raises two closely related questions for long-video understanding: **Q1.** *What makes an interaction worth computing exactly?* **Q2.** *Given such a signal, how should a limited compute budget be allocated to maximize the benefit of refinement?* Existing signals such as attention magnitude or local approximation error mainly characterize local properties of the computation, which may not reflect its value to the downstream task (Li et al., 2025a;b; Zhou et al., 2026). A visually dynamic region may be difficult to approximate yet irrelevant to the answer,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

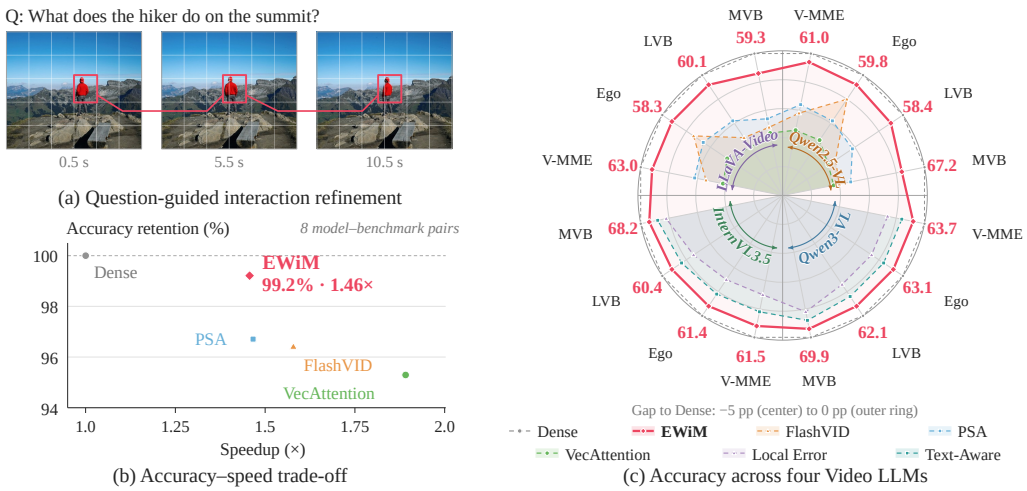


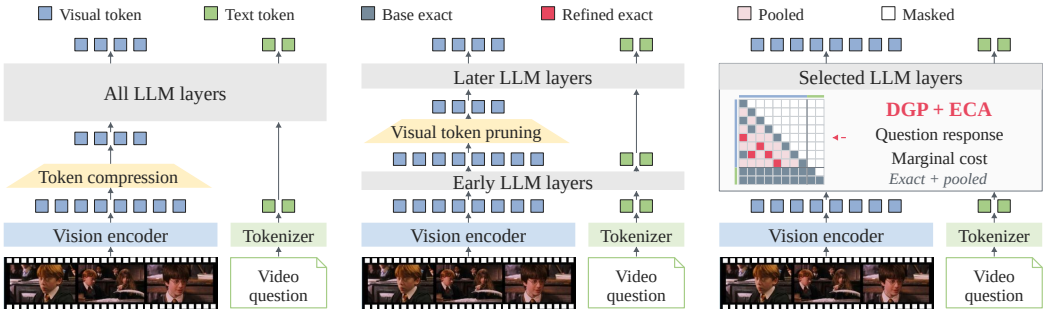
Figure 1: **EWiM overview and performance.** (a) Schematic question-guided interaction refinement; frames: Ricky Esquivel/Pexels. (b) Arithmetic-mean accuracy retention relative to Dense and geometric-mean TTFT speedup over eight main model–benchmark pairs. (c) Four model sectors, each with four benchmark axes. The shared radial scale shows accuracy differences from Dense (-5 to 0 percentage points); red labels report EWiM accuracy (%). Baseline fills cover evaluated sectors only; missing results are unplotted.

whereas a subtle interaction may be task-critical. This motivates a task-relevant proxy that better reflects the downstream benefit of exact computation.

We address these questions with **EWiM** (**E**xact **W**here **i**t **M**atters), a downstream-guided framework for allocating exact attention computation during long-video prefilling. EWiM starts from a coarse attention pass and uses lightweight local signals to form a bounded set of candidate interactions. For *Q1*, we propose **Downstream-Guided Probing** (**DGP**), which probes each candidate independently from a common coarse state by applying sampled exact corrections on its query rows. The resulting change is measured at the terminal prompt position after the next decoder layer, yielding a question-conditioned downstream signal for ranking candidates. DGP measures one-step sensitivity rather than whether a correction improves the final answer.

For *Q2*, we propose **Execution-Aware Compute Allocation** (**ECA**), which allocates the limited refinement budget according to DGP scores and marginal execution costs. Because blocks sharing query rows can reuse state updates and projection refresh, the cost of refining a block depends on which blocks have already been selected. ECA therefore updates hardware-profiled marginal costs after each selection to account for this reuse, and restores the selected blocks with full exact attention before committing the layer output. EWiM is training-free, preserves the input tokens, and leaves the backbone and decoding procedure unchanged.

We evaluate EWiM on Video-MME, EgoSchema, LongVideoBench, and MVBench (Fu et al., 2025a; Mangalam et al., 2023; Wu et al., 2024; Li et al., 2024b). Across eight overall model–benchmark comparisons using Qwen2.5-VL-7B and LLaVA-Video-7B, EWiM retains 99.2% of Dense accuracy at a $1.46\times$ geometric-mean speedup in time to first token (TTFT), including visual encoding and all refinement overhead. It also achieves the highest accuracy among the compared acceleration methods in all six Video-MME short, medium, and long comparisons. Controlled interventions show that DGP-selected refinements have higher signed answer-margin utility than those selected by local-error signals, while allocation ablations show that ECA achieves higher accuracy at comparable latency. Further experiments show transfer to Qwen3-VL-8B and InternVL3.5-8B with fixed scoring and candidate settings (Bai et al., 2025a; Wang et al., 2025), as well as composition with FlashVID through joint selection of token retention and attention refinement under common latency targets. Figure 1 summarizes the method and its measured accuracy–latency trade-offs.



(a) Before-LLM compression (b) Inner-LLM pruning (c) EWiM interaction refinement

Figure 2: **Efficiency strategies for long-video inference.** (a) Compress visual tokens before the LLM. (b) Prune tokens within the LLM. (c) EWiM retains tokens and selectively refines question-sensitive interactions. Matrix cells denote query–KV blocks; unselected eligible blocks retain pooled contributions under shared softmax normalization. The strategies are composable. Token counts and block assignments are schematic. Input illustration adapted from FlashVID (Fan et al., 2026).

2 Related Work

Video token compression. Visual-token reduction exploits temporal redundancy, feature similarity, and token importance through adaptive compression, pruning, and merging (Shen et al., 2024; Huang et al., 2025; Fu et al., 2025b; Shen et al., 2025; Shao et al., 2025). DyCoke also reduces the active KV cache during decoding (Tao et al., 2025). FlashVID combines representative-token selection with tree-based spatiotemporal merging (Fan et al., 2026). EWiM allocates exact computation among interactions over retained tokens, allowing token retention and interaction approximation to be studied separately and in composition (Figure 2).

Sparse and approximate attention. MMInference, VecAttention, and XAttention use modality-aware routing, vector-wise selection, and antidiagonal block scoring, respectively (Li et al., 2025b; Liu et al., 2026; Xu et al., 2025). PSA retains KV blocks at multiple pooling resolutions (Li et al., 2025a). In video generation, Sol-Attn combines block selection with pooled compensation (Li et al., 2026), while SVG-EAR uses lightweight probes to route exact computation by estimated compensation error per unit cost (Zhou et al., 2026). Related analyses distinguish attention concentration from output fidelity (Yang & Li, 2026) and audit changes in content influence under sparsification (Ren et al., 2026). EWiM builds on mixed exact/approximate attention, using candidate-specific next-layer responses and updating marginal costs to account for shared row computation.

Task-conditioned and feedback-guided inference. ReflectionFlow learns feedback-conditioned image refinement (Zhuo et al., 2025), and EVA learns adaptive video observation through planning, sampling, and reflection (Zhang et al., 2026). Self-ReS uses reflection-token attention for training-free visual-token selection (Pereira et al., 2025); VisReflect learns continuous visual reflections that guide attention (Shen & Elhoseiny, 2026). Vegas reuses full-attention verification signals to guide sparse drafting during self-speculative decoding (Yue et al., 2026). EWiM probes sampled block corrections through the frozen backbone's question-conditioned next-layer readout during prefilling. The resulting response measures one-step sensitivity for compute allocation; signed answer-level utility is evaluated offline.

3 Method

EWiM allocates exact attention computation during offline video–question prefilling. At selected layers, **Downstream-Guided Probing (DGP)** evaluates isolated candidate corrections through a next-layer readout, and **Execution-Aware Compute Allocation (ECA)** selects full-block refinements using their responses and conditional execution costs. Figure 3 summarizes one refinement round. Model weights, input tokens, and decoding are unchanged. Attention contributions obey the host mask, while refinement routing uses the observed question and is not prefix-invariant.

Setup. Let $X_\ell \in \mathbb{R}^{N\times D}$ enter layer ℓ of an L-layer decoder with visibility mask M. Queries, keys, and values follow the host's native normalization, positional transformations, and query-to-KV head mapping (Ainslie et al., 2023). After attention, the host's output projection, residual connections, and feed-forward operations produce the next-layer state. EWiM first constructs a provisional state $\widetilde{X}_{\ell+1}$, probes candidates without changing that state, and commits $X_{\ell+1}$ only after joint refinement. Appendix A.1 specifies the native row operations and state lifecycle. Unless indexed explicitly, attention statistics below refer to one query head at a fixed layer.

3.1 Coarse Attention and Candidate Pool

Nonoverlapping query and KV tiles I, J define an interaction block $b = (h, I, J)$ for query head h. A block is eligible for approximation only if both tiles consist entirely of visual-content tokens and every pair in $I \times J$ is visible under the host mask. Other visible interactions, including text-related and causal-boundary interactions, remain exact; masked pairs contribute zero.

Shared normalization. Write $J_i = \{j \in J : M_{ij} = 1\}$ and $a_{ij} = q_i^\top k_j/\sqrt{d}$. The exact denominator and value-weighted numerator contributed by J are

$$z_{iJ} = \sum_{j\in J_i} e^{a_{ij}}, \qquad n_{iJ} = \sum_{j\in J_i} e^{a_{ij}} v_j. \tag{1}$$

For eligible pooled blocks, we use multiplicity-corrected KV contributions (Li et al., 2025a; 2026):

$$\widetilde{z}_{iJ} = |J| e^{q_i^\top \bar{k}_J/\sqrt{d}}, \qquad \widetilde{n}_{iJ} = \widetilde{z}_{iJ}\bar{v}_J, \tag{2}$$

where means and counts use valid rows, with keys pooled after native positional transformations. Each original query is retained. The base router keeps up to k_0 eligible KV tiles exact per query tile and head. Let $\mathcal{E}_i^0$ include these tiles and all mandatory-exact regions, and let $\mathcal{P}_i^0$ contain the remaining pooled tiles. Both branches share one normalization:

$$(Z_i^0, N_i^0) = \sum_{J\in\mathcal{E}_i^0} (z_{iJ}, n_{iJ}) + \sum_{J\in\mathcal{P}_i^0} (\widetilde{z}_{iJ}, \widetilde{n}_{iJ}), \qquad o_i^0 = N_i^0/Z_i^0. \tag{3}$$

Native row operations then produce $\widetilde{X}_{\ell+1}$. We retain the coarse statistics for subsequent replacements.

Bounded candidates. DGP considers at most C_ℓ residual pooled blocks, combining ranked proposals with coverage samples. The proposal stage limits the search space; DGP and ECA determine the final selection. Appendix A.2 defines the proposal score, candidate construction, and shared row sampling.

3.2 Downstream-Guided Probing

Isolated corrections. For candidate $b = (h, I, J)$, DGP samples $r_I = \min(r, |I|)$ stratified query rows S_I. Candidates sharing I use the same samples. On each sampled row, it replaces only head h's pooled contribution:

$$\Delta z_{ib} = z_{iJ} - \widetilde{z}_{iJ}, \quad \Delta n_{ib} = n_{iJ} - \widetilde{n}_{iJ}, \quad o_i^{(b)} = \frac{N_i^0 + \Delta n_{ib}}{Z_i^0 + \Delta z_{ib}}. \tag{4}$$

Other heads retain their coarse outputs. Applying the native row operations gives $X_{\ell+1}^{(b,S)}$, which differs from $\widetilde{X}_{\ell+1}$ only on S_I. Every trial starts from the same provisional state.

Next-layer response. Let p be the terminal nonvisual prompt position after the video and question. For $\ell < L$, define

$$R_\ell(X) = [\mathcal{T}_{\ell+1}(X)]_p, \qquad s_b = \frac{\|R_\ell(X_{\ell+1}^{(b,S)}) - R_\ell(\widetilde{X}_{\ell+1})\|_2}{\|R_\ell(\widetilde{X}_{\ell+1})\|_2 + \epsilon}, \tag{5}$$

Here $\epsilon > 0$, and $\mathcal{T}_{\ell+1}$ includes native attention and feed-forward operations at p. Unlike local compensation-error routing (Zhou et al., 2026), DGP evaluates how a candidate correction propagates through this readout. The score is a sampled, one-step response magnitude; its association with answer-level utility is evaluated separately in Section 4.3. Scoring requires no answer labels or learned verifier.

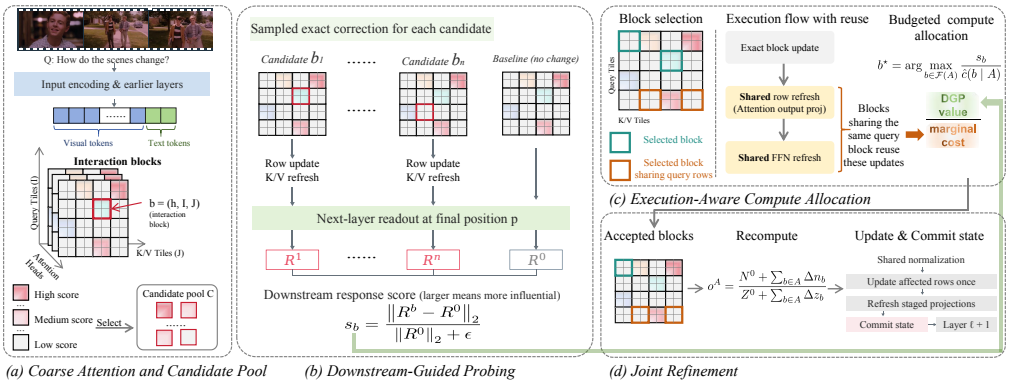


Figure 3: **EWiM inference workflow.** (a) Coarse attention constructs provisional states and candidate blocks. (b) DGP evaluates isolated sampled-row corrections through a next-layer readout with the terminal query fixed. (c) ECA keeps response scores fixed while updating marginal execution costs. (d) Accepted blocks jointly replace pooled contributions under shared normalization; affected states are refreshed before commit. Frames are from MVBench (Li et al., 2024b); blocks, samples, and selections are schematic.

Incremental evaluation. Since $p \notin S_I$, its input and next-layer query are unchanged. We share baseline terminal-row softmax statistics across candidates and update only contributions from the changed rows, whose native KV projections are refreshed. The output projection and feed-forward operations are evaluated only at p, without rerunning the remaining decoder. Appendix A.3 gives the update equations and their fixed-state conditions.

3.3 Execution-Aware Compute Allocation

Full-block refinement. Accepted blocks are recomputed over every row in I, including previously sampled entries. For $\mathcal{A}_i^{(h)} = \{b \in \mathcal{A} : h_b = h,\ i \in I_b\}$, accumulate discrepancies separately within each head:

$$(\Delta Z_i^{(h)}, \Delta N_i^{(h)}) = \sum_{b \in \mathcal{A}_i^{(h)}} (\Delta z_{ib}, \Delta n_{ib}),$$
$$o_i^{(h),\mathcal{A}} = \frac{N_i^{0,(h)} + \Delta N_i^{(h)}}{Z_i^{0,(h)} + \Delta Z_i^{(h)}}. \tag{6}$$

Head outputs are assembled before one native update per affected row, sharing output projection, feed-forward computation, and projection refresh across accepted blocks. Selected contributions are exact, while unselected proxies remain in the shared denominator.

Conditional execution cost. Let $U(\mathcal{A})$ measure the distinct query-row work induced by $\mathcal{A}$, counted once across heads. We fit

$$\hat{t}_\ell^{\mathrm{ref}}(\mathcal{A}) = \alpha_\ell |\mathcal{A}| + \beta_\ell U(\mathcal{A}) + \gamma_\ell \mathbf{1}[\mathcal{A} \neq \varnothing], \tag{7}$$
$$\hat{c}_\ell(b \mid \mathcal{A}) = \hat{t}_\ell^{\mathrm{ref}}(\mathcal{A} \cup \{b\}) - \hat{t}_\ell^{\mathrm{ref}}(\mathcal{A}). \tag{8}$$

The profile follows the deployed kernels and refresh schedule. Exact products contribute per block, whereas row refresh is shared. The additional-work budget $\mathcal{B}_\ell$ covers probing and refinement after the coarse pass. Define

$$\mathcal{F}_\ell(\mathcal{A}) = \left\{ b \in \mathcal{C}_\ell \setminus \mathcal{A} : \hat{t}_\ell^{\mathrm{probe}} + \hat{t}_\ell^{\mathrm{ref}}(\mathcal{A} \cup \{b\}) + \omega_\ell \leq \mathcal{B}_\ell \right\},$$
$$b^\star = \arg\max_{b \in \mathcal{F}_\ell(\mathcal{A})} \frac{s_b}{\hat{c}_\ell(b \mid \mathcal{A})}. \tag{9}$$

Here $\hat{t}_\ell^{\mathrm{probe}}$ estimates probing work and ω_ℓ accounts for the remaining calibrated allocation overhead. ECA updates costs after each acceptance, keeping scores fixed, and stops when the feasible

set is empty. A profile-based admission check skips unaffordable probing. Accepted corrections are committed jointly before ordinary next-layer execution. Appendix A.4 details budget accounting; the greedy rule does not assume additive answer utility or guarantee an optimal selected set.

4 EXPERIMENTS

We evaluate the accuracy–latency trade-off, the answer-level utility of DGP selections, and the execution benefit of ECA, followed by transfer and composition with token compression.

4.1 EXPERIMENTAL SETUP

Models and benchmarks. Our main experiments use frozen Qwen2.5-VL-7B-Instruct and LLaVA-Video-7B (Bai et al., 2025b; Zhang et al., 2025) on Video-MME, EgoSchema, LongVideoBench (LVB), and MVBench (MVB) (Fu et al., 2025a; Mangalam et al., 2023; Wu et al., 2024; Li et al., 2024b). We also evaluate transfer to Qwen3-VL-8B and InternVL3.5-8B (Bai et al., 2025a; Wang et al., 2025). Within each host, methods share the initial video inputs, prompts, and decoding settings.

Comparisons. We compare against Dense attention, token-compression methods FastVID and FlashVID (Shen et al., 2025; Fan et al., 2026), and attention-acceleration methods PSA, MMInference, VecAttention, and XAttention (Li et al., 2025a;b; Liu et al., 2026; Xu et al., 2025). Sol-Attn and SVG-EAR are adapted from video-generation attention to causal prefilling (Li et al., 2026; Zhou et al., 2026). Controlled variants isolate the refinement signal and allocation rule (Appendix B.2).

Evaluation. We use one H800-80GB GPU, BF16, and batch size one. The reference configuration uses $t = 64$, $k_0 = 16$, $r = 4$, and $C_\ell = 256$, with middle-layer probing. Operating points are selected on a source-disjoint development set and frozen before evaluation. We report benchmark accuracy (%) and mean time to first token (TTFT, s) on a 200-request timing panel per model–benchmark pair. TTFT spans decoded frames in host memory to first-token logits, including visual encoding and all acceleration overhead. Primary comparisons use no cross-request feature or KV-cache reuse (Appendix B).

4.2 ACCURACY AND LATENCY

Table 1 reports Video-MME short, medium, and long splits alongside overall results on four benchmarks. Across the eight overall model–benchmark comparisons, EWiM retains 99.2% of Dense accuracy at a $1.46\times$ geometric-mean speedup, with each accuracy difference within 0.8 percentage points. At comparable speedups, PSA and SVG-EAR retain 96.7% and 96.3%, respectively. VecAttention achieves a higher speedup ($1.89\times$) with lower accuracy retention (95.3%), illustrating the quality–latency trade-offs in Fig. 4.

EWiM achieves the highest accuracy among accelerated methods in all six Video-MME duration-specific comparisons, exceeding the strongest baseline by 0.9–1.9 points. On the long split, it scores 50.4% versus Dense's 49.9% on Qwen2.5-VL, and 52.3% versus 52.7% on LLaVA-Video. The corresponding strongest baselines score 49.4% and 51.4%. These duration-specific results use the same frozen configurations and prediction records as the overall evaluation.

4.3 UTILITY OF DOWNSTREAM-GUIDED PROBING

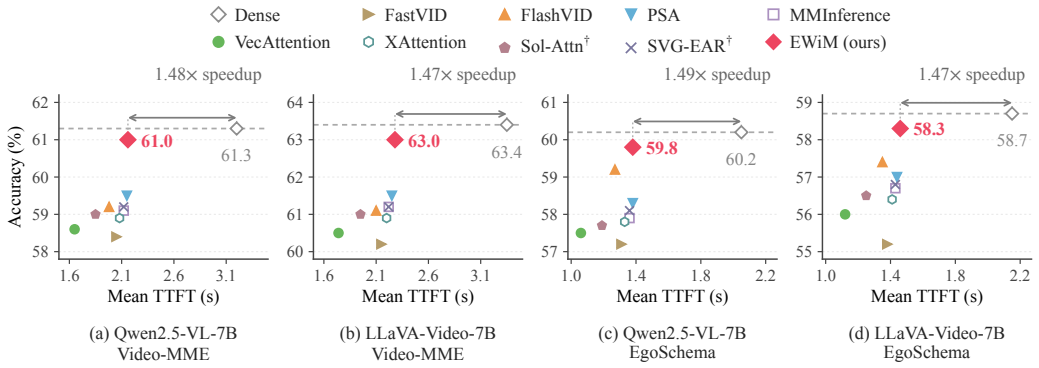
Signed refinement utility. DGP measures response magnitude, not the direction of an answer change. We therefore audit a selected block set $\mathcal{A}$ through its signed effect on the answer margin:

$$\mathcal{J}_{q\ell}(\mathcal{A}) = m_q(\mathcal{A}; \pi^0) - m_q(\varnothing; \pi^0), \tag{10}$$

where m_q is the length-normalized log-probability margin between the correct answer and a fixed competitor. Subsequent discrete routes π^0 remain fixed while dependent states are recomputed. The audit uses answer labels and full downstream reruns; neither is used to compute inference-time DGP scores (Appendix B.4).

Table 1: **Complete-system accuracy and latency.** Dataset entries are accuracy (%) / mean TTFT (s); for Video-MME duration splits (S/M/L) we report accuracy only. **Ret.** is the arithmetic mean of accuracy relative to Dense, and **Speedup** is the geometric mean of Dense TTFT divided by method TTFT, both over the eight overall model–benchmark pairs. S/M/L are not counted separately in these summaries. Bold marks the highest accuracy among accelerated methods. † denotes an adaptation from video generation.

	Qwen2.5-VL-7B							LLaVA-Video-7B							Summary	
	Video-MME				EgoSchema	LVB	MVB	Video-MME				EgoSchema	LVB	MVB	Ret.	Speedup
Method	S	M	L	Overall				S	M	L	Overall				(%)	(×)
Dense	72.6	61.4	49.9	61.3 / 3.20	60.2 / 2.05	58.9 / 3.32	68.0 / 1.70	76.2	61.3	52.7	63.4 / 3.35	58.7 / 2.15	60.5 / 3.48	60.0 / 1.82	100.0	1.00
FastVID	69.6	56.2	49.4	58.4 / 2.05	57.2 / 1.31	55.6 / 2.15	64.1 / 1.15	72.3	57.7	50.6	60.2 / 2.15	55.2 / 1.38	57.1 / 2.24	56.6 / 1.24	94.6	1.54
FlashVID	70.0	58.2	49.3	59.2 / 1.98	59.2 / 1.27	56.1 / 2.08	65.0 / 1.12	72.5	59.4	51.4	61.1 / 2.10	57.4 / 1.35	57.9 / 2.18	57.4 / 1.21	96.4	1.58
PSA	70.3	59.0	49.2	59.5 / 2.15	58.3 / 1.38	56.8 / 2.25	65.4 / 1.20	73.8	59.9	50.8	61.5 / 2.25	57.0 / 1.44	58.6 / 2.34	57.7 / 1.30	96.7	1.47
MMInference	70.0	58.5	48.8	59.1 / 2.12	57.9 / 1.36	56.4 / 2.20	65.2 / 1.18	73.5	59.6	50.5	61.2 / 2.22	56.7 / 1.43	58.2 / 2.30	57.5 / 1.28	96.2	1.49
VecAttention	69.5	58.0	48.3	58.6 / 1.65	57.5 / 1.06	55.8 / 1.72	64.8 / 0.95	72.8	58.9	49.8	60.5 / 1.74	56.0 / 1.12	57.5 / 1.80	57.2 / 1.03	95.3	1.89
XAttention	69.8	58.3	48.6	58.9 / 2.08	57.8 / 1.33	56.1 / 2.15	64.5 / 1.16	73.2	59.3	50.2	60.9 / 2.20	56.4 / 1.41	57.8 / 2.28	56.9 / 1.25	95.6	1.52
Sol-Attn†	69.9	58.5	48.6	59.0 / 1.85	57.7 / 1.19	56.2 / 1.92	64.8 / 1.08	73.3	59.4	50.3	61.0 / 1.95	56.5 / 1.25	58.0 / 2.02	57.1 / 1.17	95.8	1.68
SVG-EAR†	70.1	58.6	48.9	59.2 / 2.12	58.1 / 1.36	56.4 / 2.20	65.2 / 1.18	73.6	59.7	50.3	61.2 / 2.22	56.8 / 1.43	58.4 / 2.30	57.6 / 1.28	96.3	1.49
EWiM	**71.6**	**60.9**	**50.4**	**61.0** / 2.16	**59.8** / 1.38	**58.4** / 2.26	**67.2** / 1.20	**75.1**	**61.6**	**52.3**	**63.0** / 2.28	**58.3** / 1.46	**60.1** / 2.38	**59.3** / 1.30	**99.2**	1.46



(a) Qwen2.5-VL-7B Video-MME (b) LLaVA-Video-7B Video-MME (c) Qwen2.5-VL-7B EgoSchema (d) LLaVA-Video-7B EgoSchema

Figure 4: **Accuracy–latency operating points.** Each marker is one evaluated configuration. TTFT includes visual encoding and all acceleration overhead. Dashed lines mark Dense accuracy; arrows show Dense-to-EWiM latency changes. Accuracy axes have equal spans. † denotes adapted methods. Per-method budget sweeps are not shown.

All signals share incoming states, candidates, sampled rows, a refinement-work budget, and the ECA executor. Local Error ranks local state changes; Text-Aware Local Error additionally weights them by baseline text relevance. Table 2 shows that DGP selects sets with higher mean utility (+0.68), more positive effects (78.1%), and fewer negative effects (6.4%). Shuffling the scores removes this advantage. These measurements support DGP as a useful ranking signal without implying that every selected correction improves the answer.

Question dependence. Under matched allocation constraints, the target question's selection reaches 61.8% accuracy, versus 57.6% for another question's selection and 51.6% for random selection on the same video. All variants recompute target-question states. Figure 5 pairs this aggregate audit with a schematic illustration; the masks are not recorded routing traces. Appendix C.6 details the audit, and Appendix C.2 separately tests sampling fidelity.

4.4 Execution-Aware Allocation

Allocation benefit. Score Only ranks DGP scores without a cost adjustment; Static Cost fixes costs at the empty accepted set. ECA updates conditional costs as blocks are accepted, accounting for shared row computation. The variants use the same DGP rule and candidate-construction settings. At 2.15–2.16 s TTFT, ECA reaches 61.0% accuracy, versus 60.6% for Static Cost and 60.2% for Score Only (Fig. 6a). Its 84 accepted blocks touch 34 query tiles, compared with 78/44 and 71/52 for the controls. The blocks-per-tile ratio rises from 1.37 to 2.47 relative to Score Only, consistent

Table 2: **Signed utility of selected refinement sets.** Mean $\mathcal{J}$ is in nats per answer token. Positive and negative rates use $\epsilon = 10^{-3}$; remaining sets are neutral. States, candidates, row samples, refinement-work budget, and executor are shared.

Signal	Mean $\mathcal{J}$	Positive (%)	Negative (%)
Local Error	+0.32	58.2	13.5
Text-Aware	+0.45	64.5	11.2
DGP	**+0.68**	**78.1**	**6.4**
Shuffled DGP	-0.05	20.3	25.4

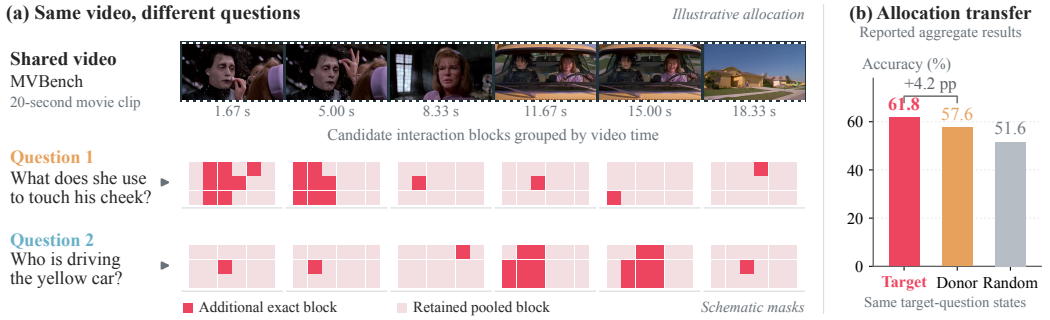


Figure 5: **Question-conditioned allocation.** (a) Two author-written questions illustrate different allocations on one MVBench clip (Li et al., 2024b). Masks are schematic additional exact blocks grouped by key time; base-exact blocks are omitted. (b) An independent aggregate audit compares target, donor, and random selections under matched constraints, with target-question states recomputed throughout.

with greater reuse. Denser Single Pass adds 132 exact blocks without probing but reaches only 60.1% at 2.17 s: block count alone does not explain the gain (Table 9).

Refinement overhead. Across 8K–64K contexts, probing, selection, exact refinement, and row refresh together take 95–1,095 ms, or 18.8–25.2% of the summed component times (Fig. 6b). ECA selection itself takes 10–105 ms. The non-negligible refresh cost supports accounting for affected query tiles when allocating computation. These component measurements are distinct from complete-system TTFT, which includes all overhead.

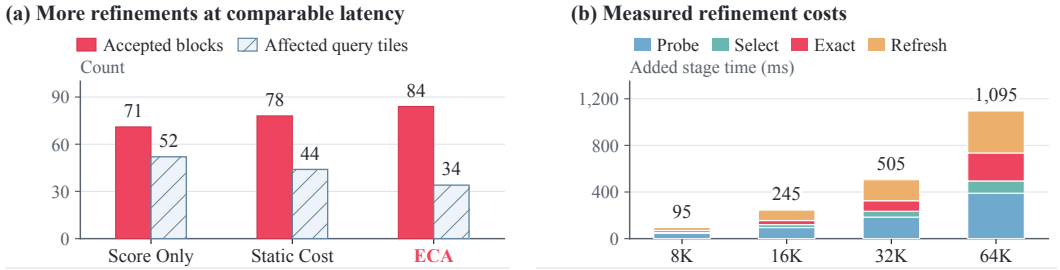
4.5 Transfer and Composition

Cross-model transfer. With scoring and candidate settings fixed, EWiM remains 0.4 points below Dense on all four combinations in Table 3, with 1.45–1.48× speedups. It improves over Text-Aware by 0.4–0.5 points. Transfer uses compatible host adapters and label-free cost recalibration; no scoring or candidate hyperparameters are retuned. Table 6 reports all four benchmarks, and Appendix C.3 examines the reference design choices.

Composition with token compression. With identical FlashVID inputs at 50% retention, EWiM improves over Dense attention by 1.2 points on both Video-MME and EgoSchema, at an additional 0.20 and 0.13 s TTFT (Table 4). Joint selection under the common latency targets allows EWiM to retain 82.5%/83.5% of visual tokens, compared with 66.0%/65.0% for Dense attention, and improves accuracy by 1.2 points on both benchmarks. This demonstrates that token retention and attention refinement can be allocated together.

5 Conclusion

We introduced **EWiM**, a training-free method for allocating exact attention computation during offline video–question prefilling. Downstream-Guided Probing ranks candidate corrections by



	Score Only	Static Cost	ECA
Acc. (%)	60.2	60.6	**61.0**
TTFT (s)	2.15	2.15	**2.16**
Blocks / tile	1.37	1.77	**2.47**

	8K	16K	32K	64K
Base (ms)	410	820	1,630	3,250
Select (ms)	10	25	50	105
Added share	18.8%	23.0%	23.7%	25.2%

Figure 6: **ECA execution benefits and refinement costs.** (a) ECA accepts more blocks across fewer query tiles at comparable TTFT (Qwen2.5-VL / Video-MME). Blocks/tile denotes their ratio. (b) Added time sums probing, selection, exact computation, and row refresh; its share uses Base plus these stages as the denominator. Component totals differ from complete-system TTFT.

Table 3: **Cross-model transfer.** Entries are accuracy (%) / mean TTFT (s). Scoring and candidate settings are fixed; host adapters preserve native attention semantics, and cost calibration uses no labels. Bold marks the best accuracy among accelerated policies.

	Qwen3-VL-8B		InternVL3.5-8B	
Policy	V-MME	EgoSchema	V-MME	EgoSchema
Dense	64.1 / 3.18	63.5 / 1.95	61.9 / 3.32	61.8 / 2.05
Coarse	56.2 / 1.86	55.7 / 1.14	55.0 / 1.95	54.9 / 1.20
Local Error	62.8 / 2.14	62.2 / 1.31	60.4 / 2.24	60.3 / 1.38
Text-Aware	63.3 / 2.15	62.7 / 1.32	61.0 / 2.26	60.9 / 1.39
EWiM	**63.7** / 2.16	**63.1** / 1.32	**61.5** / 2.28	**61.4** / 1.41

Table 4: **Composition with FlashVID on Qwen2.5-VL-7B.** Accuracy and visual-token retention (Ret.) are percentages; TTFT is in seconds. Fixed input uses 50% retention; joint selection co-selects retention and refinement under development latency targets of 2.50 s (Video-MME) and 1.60 s (EgoSchema).

	Video-MME				EgoSchema			
	Fixed input		Joint selection		Fixed input		Joint selection	
Policy	Acc.	TTFT	Ret.	Acc.	Acc.	TTFT	Ret.	Acc.
Dense	58.8	1.95	66.0	61.1	57.7	1.25	65.0	60.0
Coarse	54.2	1.70	100.0	56.0	53.2	1.09	100.0	55.0
Local Error	58.6	1.92	78.5	61.0	57.5	1.23	79.0	59.9
Text-Aware	58.9	1.94	80.0	61.4	57.8	1.24	81.0	60.3
EWiM	**60.0**	2.15	82.5	**62.3**	**58.9**	1.38	83.5	**61.2**

their sampled, question-conditioned next-layer responses, while Execution-Aware Compute Allocation updates marginal costs to account for shared row computation. Across eight main model–benchmark pairs, EWiM retains 99.2% of Dense accuracy on average at a 1.46× geometric-mean TTFT speedup, including visual encoding and all acceleration overhead. Further experiments support transfer to additional backbones and composition with token compression. Question-dependent routing limits prefix reuse across repeated questions, making independent or low-query-count inference the primary deployment setting.

AI USE STATEMENT

We used Gemini (Google) only for grammar correction and language polishing during the writing of this paper. No AI tools were involved in the research ideation, methodology, experiments, or analysis. The authors take full responsibility for all content in this manuscript.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4895–4901, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b. URL https://arxiv.org/abs/2502.13923.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Ziyang Fan, Keyu Chen, Ruilong Xing, Yulin Li, Li Jiang, and Zhuotao Tian. FlashVID: Efficient video large language models via training-free tree-based spatiotemporal token merging. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=H6rDX4w6Al.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24108–24118, 2025a.

Tianyu Fu, Tengxuan Liu, Qinghao Han, Guohao Dai, Shengen Yan, Huazhong Yang, Xuefei Ning, and Yu Wang. Framefusion: Combining similarity and importance for video token reduction on large vision language models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22654–22663. IEEE, 2025b.

Xiaohu Huang, Hao Zhou, and Kai Han. Prunevid: Visual token pruning for efficient video large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19959–19973, 2025.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024a.

Haopeng Li, Yitong Li, Junsong Chen, Tian Ye, Haozhe Liu, Jincheng Yu, Duomin Wang, Ruihua Zhang, Zeke Xie, Enze Xie, et al. Sol-attn: Accelerating video generation inference via on-the-fly attention sparsification. *arXiv preprint arXiv:2607.24027*, 2026.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22195–22206, 2024b.

Xiaolong Li, Youping Gu, Xi Lin, Weijie Wang, and Bohan Zhuang. Psa: Pyramid sparse attention for efficient video understanding and generation. *arXiv preprint arXiv:2512.04025*, 2025a.

Yucheng Li, Huiqiang Jiang, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Amir H Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, et al. Mminference: Accelerating pre-filling for long-context vlms via modality-aware permutation sparse attention. *arXiv preprint arXiv:2504.16083*, 2025b.

Anmin Liu, Ruixuan Yang, Huiqiang Jiang, Bin Lin, Minmin Sun, Yong Li, Chen Zhang, and Tao Xie. Vecattention: Vector-wise sparse attention for accelerating long context inference. *arXiv preprint arXiv:2603.29494*, 2026.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12585–12602, 2024.

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. *Advances in Neural Information Processing Systems*, 36:46212–46244, 2023.

João Pereira, Vasco Lopes, David Semedo, and João Neves. Self-res: Self-reflection in large vision-language models for long video understanding. In *2025 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–9. IEEE, 2025.

Xingyu Ren, Youran Sun, Chugang Yi, and Haizhao Yang. Understanding sparse attention selectivity in long-context foundation models via counterfactual evaluation. *arXiv preprint arXiv:2608.01676*, 2026.

Kele Shao, Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. Holitom: Holistic token merging for fast video large language models. *arXiv preprint arXiv:2505.21334*, 2025.

Leqi Shen, Guoqiang Gong, Tao He, Yifeng Zhang, pengzhang liu, Sicheng Zhao, and Guiguang Ding. FastVID: Dynamic density pruning for fast video large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2xS4VtpApy.

Xiaoqian Shen and Mohamed Elhoseiny. Visreflect: Latent visual reflection for fine-grained perception in long visual context. *arXiv preprint arXiv:2606.30288*, 2026.

Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, et al. Longvu: Spatiotemporal adaptive compression for long video-language understanding. *arXiv preprint arXiv:2410.17434*, 2024.

Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. Dycoke: Dynamic compression of tokens for fast video large language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18992–19001. IEEE, 2025.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. *Advances in Neural Information Processing Systems*, 37:28828–28857, 2024.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. *arXiv preprint arXiv:2503.16428*, 2025.

Chiwun Yang and Xiaoyu Li. Beyond sparse weights: When is attention compressible? *arXiv preprint arXiv:2608.21541*, 2026.

Yikang Yue, Yuqi Xue, and Jian Huang. Vegas: Self-speculative decoding with verification-guided sparse attention. *arXiv preprint arXiv:2602.07223*, 2026.

Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *arXiv preprint arXiv:2406.16852*, 2024.

Yaolun Zhang, Ruohui Wang, Jiahao Wang, Yepeng Tang, Xuanyu Zheng, Haonan Duan, Hao Lu, Hanming Deng, and Lewei Lu. Eva: Efficient reinforcement learning for end-to-end video agent. *arXiv preprint arXiv:2603.22918*, 2026.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Llava-video: Video instruction tuning with synthetic data, 2025. URL https://arxiv.org/abs/2410.02713.

Xuanyi Zhou, Qiuyang Mang, Shuo Yang, Haocheng Xi, Jintao Zhang, Huanzhi Mao, Joseph E Gonzalez, Kurt Keutzer, Ion Stoica, and Alvin Cheung. Svg-ear: Parameter-free linear compensation for sparse video generation via error-aware routing. *arXiv preprint arXiv:2603.08982*, 2026.

Le Zhuo, Liangbing Zhao, Sayak Paul, Yue Liao, Renrui Zhang, Yi Xin, Peng Gao, Mohamed Elhoseiny, and Hongsheng Li. From reflection to perfection: Scaling inference-time optimization for text-to-image diffusion models via reflection tuning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15329–15339. IEEE, 2025.

Supplementary Material

CONTENTS

A Additional Method Details **14**

A.1 Host Operations and State Management 14

A.2 Coarse Routing and Candidate Construction 14

A.3 Incremental Downstream Readout 14

A.4 Budgeted Selection and Execution 15

A.5 Algebraic and Numerical Consistency 15

A.6 Computational Scope and Token Compression 16

B Experimental Protocol **17**

B.1 Models, Inputs, and Evaluation 17

B.2 Baselines and Controlled Variants 17

B.3 Configuration Selection and Budget Matching 18

B.4 Signed Refinement Utility 18

B.5 Runtime and Statistical Reporting 18

C Additional Experiments and Analysis **19**

C.1 Complete Model-Transfer Results 19

C.2 Sampled-Probe Fidelity and Prediction Changes 19

C.3 Design Ablations 19

C.4 Allocation and Long-Context Execution 20

C.5 Composition with FlashVID 21

C.6 Question Dependence and Evidence Interventions 22

C.7 Repeated-Question Workloads 22

D Illustrative Analysis **23**

D.1 Question-Conditioned Allocation 23

D.2 Refinement Outcomes and Limitations 23

A Additional Method Details

A.1 Host Operations and State Management

Native row operations. Let $g(h)$ map query head h to its native KV head (Ainslie et al., 2023). The queries and keys used in $a_{ij}^{(h)} = (q_i^{(h)})^\top k_j^{(g(h))}/\sqrt{d}$ include native normalization and positional transformations. We express the post-attention computation as

$$X_{\ell+1,i} = \Phi_{\ell,i}\Big(X_{\ell,i}, \mathrm{Concat}_h\, o_i^{(h)}; F_\ell\Big), \tag{11}$$

where $\Phi_{\ell,i}$ applies the output projection, residual connections, normalization, feed-forward computation, and any fixed visual injection in native order. Auxiliary inputs F_ℓ are held fixed during a refinement round. The incremental implementation requires these operations to act independently on token rows; additional operations that couple rows or change token membership require explicit handling by the host adapter.

Prompt and mask. Tiles partition the valid token sequence without reordering it. Each evaluated query row has at least one visible key, as under the host causal mask with self-attention. Visual-content tokens exclude text, synthetic delimiters, and padding. Pooling is restricted to fully visible visual–visual tile pairs, so causal-boundary and mixed-modality interactions remain exact. The readout position p is the last valid nonvisual prompt token after the video and question, including a native generation-prefix token when present. Probing is disabled if no such position or next layer exists. Mask compliance concerns attention contributions; question-conditioned routing can change visual-prefix states across questions.

State lifecycle. The current-layer input and native Q/K/V projections remain immutable throughout probing. Candidate trials use separate temporary row states and share the same coarse baseline. Baseline next-layer projections may be staged for reuse. After selection, EWiM recomputes accepted full blocks, combines their head-wise discrepancies, and applies Φ_ℓ once per affected row. All staged projections for changed rows are refreshed against the committed state, including query projections if they were staged. Baseline and trial readouts are discarded; ordinary next-layer execution consumes only projections consistent with the finalized input.

A.2 Coarse Routing and Candidate Construction

Base routing. For each query tile and query head, the base router ranks eligible KV tiles by

$$\rho_{hIJ} = \frac{(\bar{q}_I^{(h)})^\top \bar{k}_J^{(g(h))}}{\sqrt{d}} \tag{12}$$

and selects up to k_0 for exact computation. Means use valid rows after native positional transformations. Mandatory-exact interactions are included separately and do not consume this eligible-block quota. The residual set $\mathcal{R}_\ell$ contains eligible blocks that remain pooled. Masked, mandatory-exact, and base-exact blocks are never probed.

Candidate quotas. Candidate capacity is $C'_\ell = \min(C_\ell, |\mathcal{R}_\ell|)$. A fixed ranked quota selects residual blocks by ρ_{hIJ}; a coverage quota samples without replacement from the remaining residual blocks. The branches are disjoint, and every candidate is identified by its head, query tile, and KV tile. Quotas, head allocation, and tie rules are fixed before evaluation.

Shared row sampling. For a query tile containing n_I valid rows, set $r_I = \min(r, n_I)$. Stratum $a \in \{0, \dots, r_I - 1\}$ contains row-order indices from $\lfloor a n_I / r_I \rfloor$ through $\lfloor (a+1) n_I / r_I \rfloor - 1$. DGP samples one valid row from each stratum. All candidates sharing I, including those from different heads, reuse this sample set. Their corrections remain independent because every trial starts from $\widetilde{X}_{\ell+1}$.

A.3 Incremental Downstream Readout

Fixed query, changed keys and values. A sampled replacement changes only rows S_I after the current-layer row operations. Since $p \notin S_I$, the next-layer query at p remains unchanged. Each

changed hidden row refreshes every native next-layer KV head, even when only one current-layer head was corrected. Masks, positions, token membership, and auxiliary inputs remain fixed during the trial. The readout evaluates the native next-layer block output at p, before any external final normalization.

For one next-layer query head, write its baseline terminal-row statistics as

$$Z^R = \sum_{j:M_{pj}=1} e^{a_{pj}}, \qquad N^R = \sum_{j:M_{pj}=1} e^{a_{pj}} v_j. \tag{13}$$

These statistics are computed once for the common provisional state. Let a'_{pj}, v'_j denote the values after candidate b changes S_I. The trial updates are

$$\begin{aligned} \Delta Z_b^R &= \sum_{j\in S_I: M_{pj}=1} \left(e^{a'_{pj}} - e^{a_{pj}}\right), \\ \Delta N_b^R &= \sum_{j\in S_I: M_{pj}=1} \left(e^{a'_{pj}} v'_j - e^{a_{pj}} v_j\right), \qquad o_p^{R,b} = \frac{N^R + \Delta N_b^R}{Z^R + \Delta Z_b^R}. \end{aligned} \tag{14}$$

Unchanged contributions cancel, so this equals recomputing the terminal attention row on the trial state in exact arithmetic. After assembling the head outputs, $\Phi_{\ell+1,p}$ gives the trial readout in Eq. equation 5. No other next-layer output row is required. The query is shared, but the trial's changed keys and values are not frozen.

Scope of the score. DGP computes a finite response to a sampled intervention. It neither extrapolates that response to the full tile nor estimates the sign of an answer improvement. The denominator in Eq. equation 5 is shared by candidates at the same layer and therefore does not change their ordering. Accepted blocks are evaluated again over all query rows; sampled products are not committed as partial refinements. Multiple accepted corrections can interact through normalization and downstream nonlinear operations, so their joint effect need not equal the sum of their isolated responses.

A.4 Budgeted Selection and Execution

Row sharing. For the valid-row convention, $U(\mathcal{A}) = |\bigcup_{b\in\mathcal{A}} I_b|$. Heads sharing the same query rows contribute separately to $|\mathcal{A}|$ but only once to U. The conditional cost expands to

$$\hat{c}_\ell(b \mid \mathcal{A}) = \alpha_\ell + \beta_\ell[U(\mathcal{A}\cup\{b\}) - U(\mathcal{A})] + \gamma_\ell \mathbf{1}[\mathcal{A} = \varnothing]. \tag{15}$$

The first term charges the new head-block product; the remaining terms charge newly affected rows and initial execution overhead. Profiling uses the deployed batching and refresh schedule. The selection rule is defined for positive fitted marginal costs.

Budget accounting. The budget $\mathcal{B}_\ell$ covers additional work after coarse execution and uses the same time units as the profile. Candidate construction, shared readout preparation, sampled trials, and allocator overhead must be included in $\hat{t}_\ell^{\text{probe}} + \omega_\ell$. The refinement profile covers accepted exact products and affected-row updates. A native projection pass belongs to ordinary execution once, even when staged early; repeated projections and discarded readout work are overhead. Probing costs remain charged when no candidate is accepted. The constraint is profile-based and does not impose a hard wall-clock bound.

Admission and selection. Before probing, a shape-based estimate checks whether the budget can cover the fixed overhead and a singleton refinement. If it cannot, the layer returns its coarse state. Otherwise, Algorithm 1 computes candidate scores once and repeatedly selects from the feasible set in Eq. equation 9. The cost changes as query rows become reusable; scores remain fixed. Zero refinement budget bypasses probing.

A.5 Algebraic and Numerical Consistency

Replacement identity. Fix a layer's Q/K/V, mask, and query head. Each accepted block is a distinct pooled block from the baseline. Expanding Eq. equation 3, subtracting its proxy, and adding

Algorithm 1 One EWiM refinement round

Require: Committed X_ℓ, frozen configuration, host adapter, profile, and budget $\mathcal{B}_\ell$
Compute coarse statistics, provisional state $\widetilde{X}_{\ell+1}$, and residual set $\mathcal{R}_\ell$.
if probing is disabled, the readout is unavailable, or $\mathcal{R}_\ell = \varnothing$ **then**
  return $\widetilde{X}_{\ell+1}$.
end if
if the profile predicts no affordable probe-and-refine round **then**
  return $\widetilde{X}_{\ell+1}$.
end if
Construct $\mathcal{C}_\ell$ and shared samples S_I; prepare baseline readout statistics.
Evaluate independent sampled trials in microbatches and compute s_b.
$\mathcal{A} \leftarrow \varnothing$.
while $\mathcal{F}_\ell(\mathcal{A}) \neq \varnothing$ **do**
  $b^\star \leftarrow \arg\max_{b \in \mathcal{F}_\ell(\mathcal{A})} s_b / \hat{c}_\ell(b \mid \mathcal{A})$.
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{b^\star\}$; update conditional costs and feasibility.
end while
Recompute accepted full blocks; combine discrepancies within each row and head.
Apply native row updates, or retain the coarse state if $\mathcal{A} = \varnothing$.
Refresh staged projections on changed rows and discard temporary readouts.
Commit $X_{\ell+1}$ and its consistent projections; record all incurred work.

its exact contribution leaves

$$(Z_i^{\mathcal{A}}, N_i^{\mathcal{A}}) = \sum_{J \in \mathcal{E}_i^0 \cup \mathcal{A}_i} (z_{iJ}, n_{iJ}) + \sum_{J \in \mathcal{P}_i^0 \setminus \mathcal{A}_i} (\widetilde{z}_{iJ}, \widetilde{n}_{iJ}), \tag{16}$$

where $\mathcal{A}_i$ denotes the selected KV tiles for this fixed row and head. This is precisely the mixed attention policy represented by Eq. equation 6. Distinct blocks prevent duplicate subtraction, and each remaining denominator contribution is positive for finite logits and a nonempty visible tile. For a single replacement,

$$o_i^{(b)} - o_i^0 = \frac{\Delta n_{ib} - o_i^0 \Delta z_{ib}}{Z_i^0 + \Delta z_{ib}}. \tag{17}$$

Thus, discrepancies must be combined before normalization; separately normalized output corrections cannot generally be added. Replacing all pooled blocks recovers dense attention on the fixed layer input. Recovering the dense model's complete trajectory additionally requires exact attention at every layer.

Shared exponent scale. The exponential expressions above specify the algebra. Stable execution uses a common exponent scale for the baseline, exact terms, and removed proxies, with FP32 accumulation (Dao et al., 2022). If stored statistics use scale m_0 and an update uses scale m, both stored numerator and denominator are first rescaled by $e^{m_0 - m}$. Readout differences and norms are computed before terminal low-precision storage. Positive denominators in exact arithmetic do not rule out finite-precision cancellation.

A.6 Computational Scope and Token Compression

For a predominantly visual sequence of N_v tokens, tile size t, and H_q query heads, dense tile-proposal construction costs $O(H_q N_v^2 d/t^2)$ and row-to-pooled-KV attention costs $O(H_q N_v^2 d/t)$, in addition to exact regions and native tokenwise operations. Bounding C_ℓ limits candidate trials; it does not make the whole prefill linear. Baseline readout preparation also depends on sequence length. Ordinary token-indexed KV storage is retained, while provisional states, routing statistics, and trial buffers add workspace.

EWiM can operate on a fixed token-compressed representation (Fan et al., 2026); tiles, candidates, and cost estimates are constructed for the resulting sequence. If a compression pipeline also prunes tokens inside the decoder, the fixed-membership condition of Appendix A.3 must hold across each probed transition. A pruning decision that changes between a baseline and trial requires accounting beyond the changed-row update in Eq. equation 14.

Table 5: **Evaluation inventory.** N is the formatted prompt length, excluding padding and generated answers. Video-MME duration splits partition the full split by video length.

Dataset	Split	Questions	Median N	Range of N
Video-MME (Short)	Full	900	5,240	4,210–8,960
Video-MME (Medium)	Full	900	16,820	8,960–28,450
Video-MME (Long)	Full	900	42,580	28,450–63,850
Video-MME (Overall)	Full	2,700	—	4,210–63,850
EgoSchema	Full	5,031	11,860	9,860–14,150
LongVideoBench	Validation	1,337	24,150	8,400–58,200
MVBench	20 tasks	4,000	6,800	2,100–12,400

B Experimental Protocol

B.1 Models, Inputs, and Evaluation

Models and benchmarks. Main experiments use frozen Qwen2.5-VL-7B-Instruct and LLaVA-Video-7B (Bai et al., 2025b; Zhang et al., 2025); transfer uses Qwen3-VL-8B and InternVL3.5-8B (Bai et al., 2025a; Wang et al., 2025). We evaluate the full Video-MME split without subtitles, the full EgoSchema benchmark of 5,031 five-way questions over three-minute clips, the LongVideoBench validation split with interleaved subtitles, and all 20 MVBench tasks without subtitles (Fu et al., 2025a; Mangalam et al., 2023; Wu et al., 2024; Li et al., 2024b). EWiM modifies prefilling only; model parameters and autoregressive decoding remain unchanged.

Common inputs and scoring. Within a host, methods share frame sampling, resolution limits, prompts, context and output limits, and the answer parser. Token-compression methods act on the same initial representation; attention-only methods retain the formatted sequence and positional indices. Table 5 reports the input inventory. Accuracy follows the official benchmark protocol. Invalid predictions count as incorrect; missing inputs and execution failures are logged separately. All subgroup results use the same prediction records as the aggregate evaluation.

B.2 Baselines and Controlled Variants

External methods. FastVID and FlashVID compress visual tokens (Shen et al., 2025; Fan et al., 2026); PSA, MMInference, VecAttention, and XAttention reduce attention computation (Li et al., 2025a;b; Liu et al., 2026; Xu et al., 2025). Sol-Attn and SVG-EAR are adapted from video generation (Li et al., 2026; Zhou et al., 2026), retaining their routing principles while respecting the host's causal mask, positional encoding, and attention normalization. These adaptations use neither answer labels nor DGP scores. Reported results apply to the causal adaptations, not the original video-generation implementations.

Local Error. For candidate $b = (h, I, J)$ and sampled query rows S_I, let $\Delta X_i^{(b)} = X_{\ell+1,i}^{(b,S)} - \widetilde{X}_{\ell+1,i}$ denote the state change after the current layer. Local Error ranks candidates by

$$e_b = \left(\frac{1}{|S_I|} \sum_{i \in S_I} \|\Delta X_i^{(b)}\|_2^2 \right)^{1/2}. \tag{18}$$

This score measures local correction magnitude without evaluating the next-layer response.

Text-Aware Local Error. This control weights e_b by baseline next-layer attention from the final text position p to query tile I:

$$w_b = \frac{1}{H_{\ell+1}} \sum_{g=1}^{H_{\ell+1}} \sum_{i \in I} A_{pi}^{0,(g)}, \qquad s_b^{\mathrm{TA}} = w_b e_b. \tag{19}$$

It incorporates question-conditioned relevance without propagating the candidate-specific correction. Tables abbreviate it as Text-Aware. Shuffled DGP permutes DGP scores among candidates

within a question and layer, preserving the score distribution while breaking candidate–score correspondence.

Allocation controls. Coarse disables probing and refinement. Denser Single Pass increases the base-exact set without probing. Score Only ranks s_b without cost adjustment. Static Cost keeps $\hat{c}_\ell(b \mid \varnothing)$ fixed throughout selection. ECA updates $\hat{c}_\ell(b \mid \mathcal{A})$ after acceptance to account for shared affected-row computation. Block counts in Table 9 exclude the common coarse pass; Denser Single Pass reports additional base-exact blocks.

B.3 Configuration Selection and Budget Matching

Development selection. Configurations are selected on 200 source-disjoint development videos and frozen before evaluation. Answer labels are used to select quality–latency operating points, while hardware profiling uses no labels. The reference uses $t = 64$, $k_0 = 16$, $C_\ell = 256$, a 50:50 ranked-to-coverage candidate ratio, $r = 4$, probing microbatch size $G = 32$, and a fixed middle-layer schedule. Transfer retains these scoring and candidate settings, with compatible host adapters and label-free cost recalibration.

Complete-system comparisons. Development latency targets are $\{0.50, 0.65, 0.80\}$ times Dense mean TTFT, with 0.65 used for the primary comparison. EWiM searches $k_0 \in \{8, 16, 32\}$ and $\lambda_B \in \{0, 0.1, 0.25, 0.5, 1\}$; external methods vary their native retention, sparsity, or compute controls. At each target, we select the feasible measured configuration with the highest development accuracy, breaking ties by latency. Infeasible targets are omitted. Evaluation latency is measured independently and need not equal the development target. No unmeasured operating points are inferred by interpolation.

Mechanism comparisons. Signal diagnostics share the incoming state, coarse policy, candidates, sampled rows, refinement-work budget, and ECA executor; only the signal changes. Allocation comparisons use the same DGP rule and candidate-construction settings. Fixed-state comparisons isolate layer-local decisions; independently executed policies can produce different subsequent states after committing different refinements. Complete-system TTFT includes signal acquisition and allocation overhead for every policy.

B.4 Signed Refinement Utility

For question q, let y_q^+ and y_q^- be correct and competing answer strings, selected before intervention and held fixed. With $\bar{L}_q(y; \mathcal{A}, \pi^0) = |y|^{-1} \log P(y \mid v, q; \mathcal{A}, \pi^0)$, where $|y|$ is the answer's token count, define

$$m_q(\mathcal{A}; \pi^0) = \bar{L}_q(y_q^+; \mathcal{A}, \pi^0) - \bar{L}_q(y_q^-; \mathcal{A}, \pi^0). \tag{20}$$

Equation 10 compares a jointly restored block set with the same unrefined incoming state. Later discrete routes π^0 are fixed, while all dependent hidden states and caches are recomputed. Joint utility is measured directly rather than obtained by summing single-block effects. Positive and negative sets satisfy $\mathcal{J} > 10^{-3}$ and $\mathcal{J} < -10^{-3}$; the remainder are neutral. Units are nats per answer token. The audit uses labels and full downstream reruns; neither is used to compute inference-time DGP scores.

B.5 Runtime and Statistical Reporting

Timing boundary. All measurements use one H800-80GB, BF16, and batch size one. TTFT starts with decoded frames in host memory and ends at first-token logits. It includes input construction and transfer, visual encoding, compression when applicable, prefill, candidate construction, probing, selection, exact refinement, and state refresh. Video-file decoding and subsequent autoregressive generation are excluded. Primary comparisons reuse neither visual features nor KV caches across requests; Appendix C.7 explicitly permits compatible reuse.

Timing and uncertainty. For each model–benchmark pair, the timing panel contains 200 requests spanning the formatted sequence-length range. After warm-up, each request is timed repeatedly with

Table 6: **Complete model-transfer results.** Entries are accuracy (%) / mean TTFT (s); Video-MME duration splits (S/M/L) report accuracy only. Scoring and candidate settings are fixed across models; only the hardware-dependent cost model is recalibrated without labels. **Ret.** is the arithmetic mean of accuracy retention relative to Dense over the four overall benchmark results; **Spd.** is the geometric-mean speedup.

	Video-MME				EgoSchema	LVB	MVB	Average	
Policy	S	M	L	Overall				Ret. (%)	Spd. (×)
Qwen3-VL-8B									
Dense	75.4	64.2	52.7	64.1 / 3.18	63.5 / 1.95	62.5 / 3.30	70.2 / 1.69	100.0	1.00
Coarse	67.2	55.7	45.7	56.2 / 1.86	55.7 / 1.14	54.8 / 1.93	64.6 / 0.99	88.8	1.71
Local Error	73.8	62.3	52.3	62.8 / 2.14	62.2 / 1.31	61.2 / 2.22	69.3 / 1.14	98.1	1.49
Text-Aware	74.3	62.8	52.8	63.3 / 2.15	62.7 / 1.32	61.7 / 2.23	69.6 / 1.14	98.8	1.48
EWiM	**74.9**	**63.3**	**52.9**	**63.7** / 2.16	**63.1** / 1.32	**62.1** / 2.24	**69.9** / 1.15	**99.4**	1.47
InternVL3.5-8B									
Dense	73.2	62.0	50.5	61.9 / 3.32	61.8 / 2.05	60.8 / 3.45	68.5 / 1.76	100.0	1.00
Coarse	66.0	54.5	44.5	55.0 / 1.95	54.9 / 1.20	53.4 / 2.02	63.0 / 1.03	89.4	1.70
Local Error	71.4	59.9	49.9	60.4 / 2.24	60.3 / 1.38	59.5 / 2.33	67.6 / 1.19	97.9	1.48
Text-Aware	72.0	60.5	50.5	61.0 / 2.26	60.9 / 1.39	60.0 / 2.35	67.9 / 1.20	98.7	1.47
EWiM	**72.7**	**61.1**	**50.7**	**61.5** / 2.28	**61.4** / 1.41	**60.4** / 2.37	**68.2** / 1.21	**99.4**	1.46

device synchronization at request boundaries; tables report mean TTFT over this panel. Development and evaluation timings include all method-specific scoring, recovery, and refresh work. Where per-question correctness is available, paired accuracy intervals bootstrap source videos, retaining all questions from each sampled video. Routing-seed variation and timing variation are assessed separately from paired accuracy uncertainty.

C Additional Experiments and Analysis

C.1 Complete Model-Transfer Results

Table 6 reports the complete transfer results, including Video-MME S/M/L and overall scores on all four benchmarks. EWiM remains within 0.4 points of Dense on the eight overall model–benchmark comparisons, retaining 99.4% of Dense accuracy on each transfer host. It also achieves the highest accuracy among the evaluated accelerated policies in each duration group. All configurations are frozen before evaluation and are not retuned for individual duration splits.

C.2 Sampled-Probe Fidelity and Prediction Changes

We audit 128 source-distinct questions per benchmark across sequence lengths and multiple probed layers. For each unshuffled signal, sampled-row scores are compared with that signal's full-query-tile counterpart. DGP achieves Kendall's $\tau_b = 0.71$–0.74, compared with 0.12–0.14 for Local Error and 0.18–0.19 for Text-Aware; shuffled scores have near-zero agreement (Table 7). This measures sampling fidelity, separately from the signed answer utility in Table 2. On LVB, increasing r from 1 to 2, 4, and 8 yields $\tau_b = 0.58, 0.68, 0.74, 0.78$, respectively, with a smaller gain beyond four rows.

Prediction changes. On a separate audit of 500 LVB questions, EWiM repairs 44 Coarse errors and introduces 10 regressions, versus 35/6 for Local Error and 38/7 for Text-Aware. Its net gain is therefore 34 questions (6.8 points), compared with 29 and 31. EWiM has the largest net improvement but also more regressions. These complete-run outcomes complement the controlled utility audit; they do not estimate individual-block effects.

C.3 Design Ablations

Table 8 varies one component at a time on Qwen2.5-VL / Video-MME, retaining the reference settings for the others. TTFT includes all overhead.

Table 7: **Sampled-probe fidelity.** Kendall's τ_b with 95% bootstrap intervals on 128 source-distinct questions per benchmark, spanning sequence lengths and multiple probed layers. Each unshuffled signal is compared with its own full-query-tile counterpart; Shuffled DGP is a null control.

Signal	Video-MME	LVB	EgoSchema
Local Error	0.13 [0.09, 0.17]	0.12 [0.08, 0.16]	0.14 [0.10, 0.18]
Text-Aware	0.19 [0.15, 0.23]	0.18 [0.14, 0.22]	0.19 [0.15, 0.23]
DGP	**0.73** [0.68, 0.78]	**0.74** [0.69, 0.79]	**0.71** [0.66, 0.76]
Shuffled DGP	0.00 [−0.03, 0.04]	0.01 [−0.03, 0.05]	−0.01 [−0.04, 0.03]

Table 8: **Candidate and probing design on Qwen2.5-VL / Video-MME.** TTFT includes all overhead. Blocks counts accepted refinements, excluding base-exact blocks. Block counts across different tile sizes do not measure equal work. Bold marks the reference configuration.

Variant	Accuracy (%)	TTFT (s)	Blocks
Candidate source and pool size			
EWiM reference	**61.0**	**2.16**	**84**
Ranked only	60.4	2.05	42
Coverage only	58.8	2.05	42
Ranked:coverage = 75:25	60.9	2.14	79
Ranked:coverage = 25:75	60.2	2.13	76
$C_\ell = 128$	60.7	2.10	71
$C_\ell = 512$	61.1	2.29	89
Sampled query rows per tile (r)			
$r = 1$	60.2	2.10	84
$r = 2$	60.7	2.12	84
$r = 8$	61.2	2.28	84
Probing layer schedule			
Early layers	60.1	2.15	84
Late layers	60.5	2.17	84
Early + middle layers	61.1	2.31	112
All layers	61.1	2.48	146
Tile size (t) and base-exact count (k_0)			
$t = 32$	60.6	2.31	157
$t = 128$	60.3	2.08	46
$k_0 = 8$	60.4	2.07	96
$k_0 = 32$	61.1	2.29	71

Candidates and sampled rows. Ranked-only and coverage-only variants reach 60.4% and 58.8%, compared with 61.0% for the mixed reference. Both also halve the accepted-block count and reduce TTFT to 2.05 s, so these removals do not isolate candidate diversity at fixed compute. Expanding the pool from 256 to 512 adds 0.1 accuracy points at 0.13 s additional TTFT. Increasing r from 1 to 8 raises accuracy from 60.2% to 61.2% and TTFT from 2.10 to 2.28 s; the reference $r = 4$ reaches 61.0% at 2.16 s.

Layers and tile granularity. Middle-layer probing outperforms early- and late-layer schedules at similar latency. Adding early layers or probing all layers gains 0.1 points while raising TTFT to 2.31 or 2.48 s. Smaller tiles ($t = 32$) reach 60.6% at 2.31 s, while larger tiles ($t = 128$) reach 60.3% at 2.08 s. These settings change both candidate granularity and execution cost; their block counts are not comparable measures of work. Increasing k_0 to 32 reaches 61.1% at 2.29 s. The reference balances quality and latency among these measured configurations, without implying an optimum beyond the tested settings.

C.4 Allocation and Long-Context Execution

Table 9 gives the controls underlying Fig. 6a. Relative to Score Only, ECA accepts 13 more blocks while touching 18 fewer query tiles at almost unchanged TTFT. Denser Single Pass uses 57% more

Table 9: **Allocation strategies on Qwen2.5-VL / Video-MME.** TTFT includes all overhead. Blocks counts additional exact blocks beyond Coarse; Q Tiles counts the distinct query tiles they touch. Denser Single Pass adds base-exact blocks without probing.

Policy	Accuracy (%)	TTFT (s)	Blocks	Q Tiles
Coarse	54.3	1.88	0	0
Denser Single Pass	60.1	2.17	132	58
Score Only	60.2	2.15	71	52
Static Cost	60.6	2.15	78	44
ECA	**61.0**	2.16	84	34

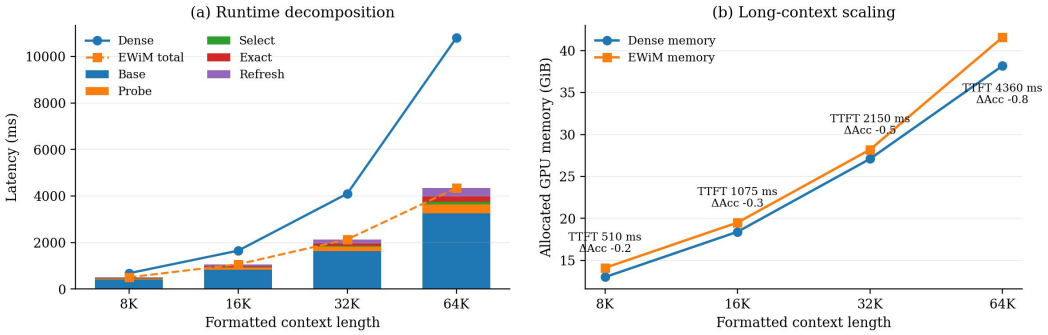


Figure 7: **Runtime and long-context scaling.** (a) Profiled component times for the coarse pass, probing, selection, exact refinement, and row refresh. (b–d) Independently measured TTFT, allocated GPU memory, and accuracy difference from Dense across 8K–64K formatted contexts. Memory is in GiB; accuracy differences are in percentage points.

additional exact blocks than ECA but has lower accuracy; this compares block counts, not total FLOPs. Together, the controls support using both downstream scores and conditional execution costs.

Across 8K–64K contexts, probing, selection, exact refinement, and row refresh contribute 18.8–25.2% of the summed component times. This denominator includes the coarse pass and is distinct from independently measured TTFT. At 64K, allocated GPU memory is 41.6 GiB for EWiM versus 38.2 GiB for Dense, with an accuracy gap of 0.8 points (Fig. 7). EWiM reduces attention computation while retaining the input sequence and adding workspace for probing and refinement; these results do not indicate a reduction in the ordinary KV-cache footprint.

C.5 Composition with FlashVID

The fixed-input comparison in Table 4 holds the FlashVID representation constant at 50% token retention (Fan et al., 2026). Relative to Dense attention on those tokens, EWiM improves accuracy by 1.2 points on both benchmarks, with TTFT increasing from 1.95 to 2.15 s on Video-MME and from 1.25 to 1.38 s on EgoSchema. This comparison isolates the attention policy from the choice of retained representation, but does not match latency.

Joint selection varies retention and attention computation under development targets of 2.50 s and 1.60 s, approximately 0.78 times uncompressed Dense TTFT. EWiM selects 82.5%/83.5% retention and reaches 62.3%/61.2% accuracy, compared with 66.0%/65.0% and 61.1%/60.0% for the Dense-attention policy. Coarse retains all tokens but has lower accuracy. Thus, retaining more tokens alone is insufficient; the attention policy also matters. EWiM's accuracy exceeds uncompressed Dense by 1.0 point on each benchmark, an empirical task result rather than evidence that refinement restores information discarded by merging. Development targets specify configuration selection and should not be read as per-configuration measured evaluation times.

Table 10: **Visual-evidence interventions.** Reported performance (%) for edits to local actions, temporal order, and individual critical events, together with the stability measure for answer-preserving sham edits. Bold marks the best accelerated policy in the first three columns.

Policy	Local pair	Order pair	Single	Sham
Dense	65.5	62.0	64.2	99.0
Coarse	42.0	40.5	50.1	95.5
Local Error	47.5	46.0	55.2	98.2
Text-Aware	49.0	48.0	56.4	98.4
EWiM	**54.5**	**52.5**	**61.8**	98.5

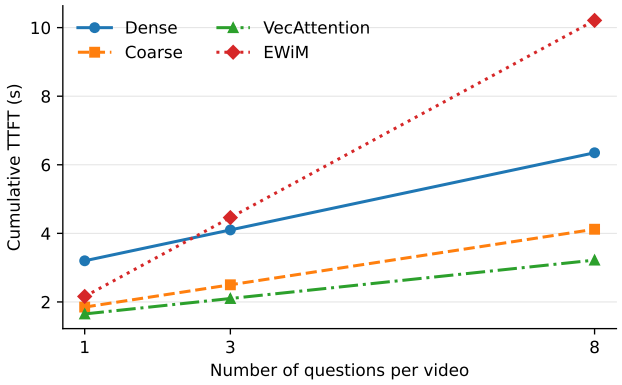


Figure 8: **Repeated questions on a cached video.** Cumulative first-token latency for one, three, and eight questions per source video, allowing compatible question-independent reuse for every method. Lines connect measured workloads.

C.6 Question Dependence and Evidence Interventions

Allocation transfer. For two questions about different evidence in one video, we compare target-question selection, donor-question selection, and random selection. Candidates, sampled rows, query-tile/head quotas, and refinement budgets are matched. Only block identities transfer; target-question states are recomputed without donor states or caches. Target selection reaches 61.8% accuracy, compared with 57.6% and 51.6% for donor and random selection. The 4.2-point donor penalty supports conditioning allocation on the question under these constraints.

Visual interventions. Verified edits change a local action, temporal order, or a critical event while holding the question fixed; matched sham edits preserve the answer. EWiM outperforms both local refinement controls in all three evidence-change categories, while remaining below Dense (Table 10). Its sham score is similar to those of the local controls. This audit complements allocation transfer by changing the evidence rather than the question.

C.7 Repeated-Question Workloads

We measure cumulative first-token latency for one, three, and eight independent questions about the same source video. Each method can reuse compatible question-independent computation, while EWiM recomputes question-conditioned scores and refinement decisions. For M questions, $T(M) = T_{\text{first}} + \sum_{j=2}^{M} T_j$ is measured from observed executions, not extrapolated from single-request TTFT. EWiM's cumulative latency grows faster than VecAttention's (Fig. 8). Its single-request benefit therefore does not establish an advantage for workloads with extensive video-prefix reuse.

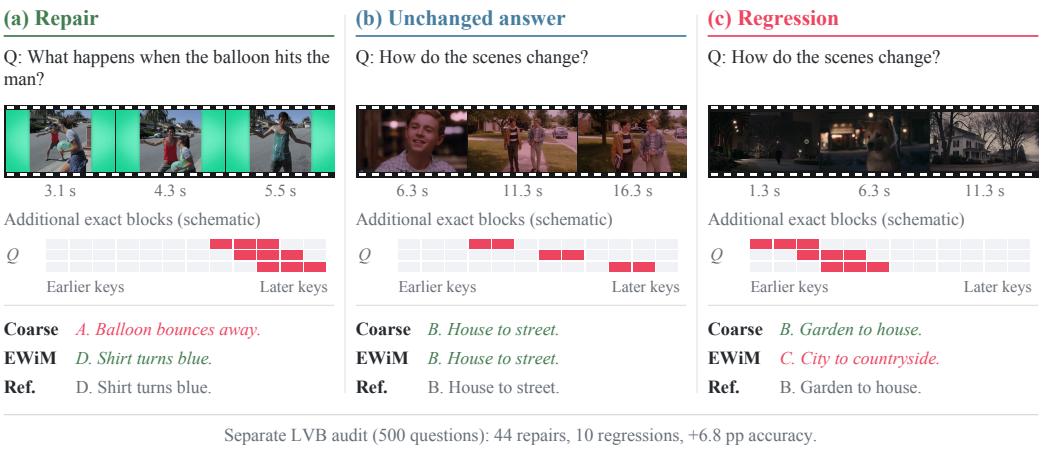


Figure 9: **Illustrative refinement outcomes.** MVBench frames illustrate a repair, an unchanged answer, and a regression (Li et al., 2024b). Questions and reference answers are shortened from benchmark annotations; predictions and additional exact-block masks are schematic, not recorded outputs. Base-exact blocks are omitted. The bottom row reports a separate LVB audit, independent of the illustrated clips.

D Illustrative Analysis

D.1 Question-Conditioned Allocation

Figure 5 illustrates how questions about one video can favor exact computation in different temporal regions. The masks are schematic, not recorded routing decisions. The target-, donor-, and random-allocation results come from the independent quantitative audit in Appendix C.6; they do not measure the illustrated clip.

D.2 Refinement Outcomes and Limitations

Figure 9 illustrates a repaired answer, an unchanged answer, and a regression using MVBench frames and reference annotations. Coarse/EWiM predictions and allocation masks are schematic. The aggregate LVB results below the examples are independent of these clips.

DGP measures response magnitude rather than its direction relative to correctness. A large response can accompany a harmful correction, and a changed readout need not change the predicted answer. The signed-utility and prediction-change audits quantify these outcomes; the illustrations explain the distinction without serving as evidence of measured case-level behavior.